Generalized Linear Model

Contents

Introduction

Generalized Linear Models (GLM) offer a powerful and flexible statistical methodology that generalizes traditional linear models to accommodate various types of data and distributions. These models are essential for understanding complex relationships in data where the response distribution does not necessarily follow normality. This section is mainly based on [**?**].

GLMs are applied using distributions from the exponential family, leveraging their beneficial mathematical properties, such as simplification in the calculation of expectation and variance. Below is the definition of this family of distributions.

A univariate random variable Y, which depends on a single parameter θ, belongs to the *exponential family* if its density function can be expressed as

$$f(y, \theta) = \exp\left\{a(y)b(\theta) + c(\theta) + d(y)\right\},$$

where $a(y)$, $b(\theta)$, $c(\theta)$, and $d(y)$ are known functions that define the structure of the distribution. Additionally, if $a(y) = y$, the distribution is said to have the *canonical form*.

A GLM analyzes the relationship between a dependent variable and p explanatory variables based on a random sample with the following characteristics:

1. The variables $Y_1, \ldots, Y_n$ follow the same distribution from the exponential family in the canonical form with a single parameter of interest θ_i for each i.
2. The relationship between $\eta_i = x_i^T \beta$ and $\mu_i = \mathrm{E}(Y_i)$ is defined by a monotonic and differentiable function, g, called the *link function*,
$$g(\mu_i) = x_i^T \beta. \tag{1}$$

Model Estimation

The estimation process for GLMs involves the following steps:

1. Initialize with an initial value for the model parameters.
2. In each iteration, update the matrices W and z based on the estimated values of $\beta^{(m-1)}$.
3. Recalculate the parameter vector β using the formula
$$\beta^{(m)} = (\mathbf{X}^T W \mathbf{X})^{-1} \mathbf{X}^T W z.$$

Where:

- $w_{ii} = \frac{1}{\mathrm{Var}(Y_i)} \left(\frac{\partial \mu_i}{\partial \eta_i}\right)^2$.

- $z_i = \sum_{k=1}^{p} x_{ik}\beta_k^{(m-1)} + (y_i - \mu_i)\frac{\partial \eta_i}{\partial \mu_i}$.

Simulating data

First, we load the necessary libraries for data visualization and analysis.

```r
library(ggplot2)
library(gridExtra)
library(caret) # confusion matrix
```

Next, we generate the simulated data for the GLM analysis. We create a binary response variable using a logistic function.

```r
n <- 300
x <- runif(n, 0, 1)
x <- x[order(x)]

# Calculate probability using logistic function
prob <- 1 / (1 + exp(-(-8 + 16 * x)))  # Logistic function

# Generate binary response variable
y <- rbinom(n, 1, prob)

# Create a data frame
data <- data.frame(x = x, y = y, prob = prob)
```

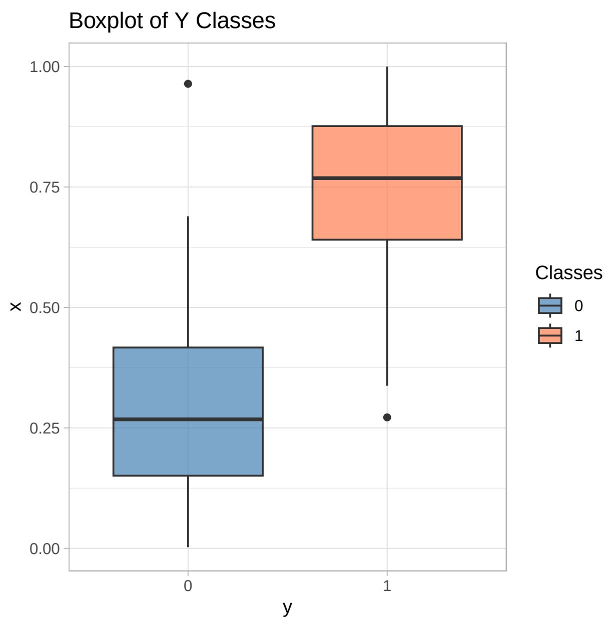
We visualize the generated data using boxplots and scatter plots to understand the distribution and relationship of the variables.

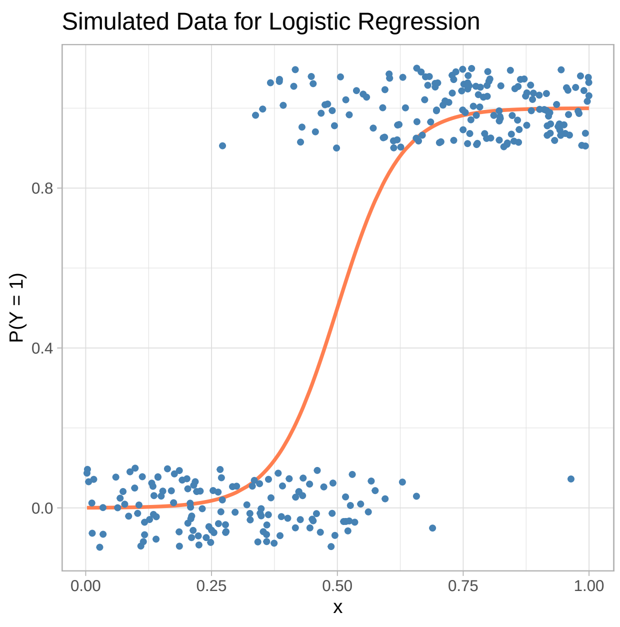
```r
colores <- c(adjustcolor("steelblue", alpha.f = 0.7),
             adjustcolor("coral", alpha.f = 0.7))

plot1 <- ggplot(data, aes(x = factor(y), y = x, fill = factor(y))) +
  geom_boxplot() +
  labs(x = "y",
       y = "x",
       title = "Boxplot of Y Classes",
       color = NULL) +
  scale_fill_manual(values = colores, name = "Classes") +
  theme_light()

plot0 <- ggplot(data = data) +
  geom_line(aes(x = x, y = prob), color = "coral", linewidth = 1) +
  geom_jitter(aes(x = x, y = y),
              position = position_jitter(height = 0.1), shape = 20, size = 2, color = "steelblue") +
  labs(title = "Simulated Data for Logistic Regression",
       x = "x",
       y = "P(Y = 1)",
       color = NULL) +
  theme_light() +
  theme(plot.title = element_text(size = 14)) +
  guides(color = "none")

newplot <- grid.arrange(plot1, plot0, nrow = 1, ncol = 2)
```





Fitting the GLM Model

We fit a GLM to the data using the `glm` function in R.

```r
glm_model <- glm(y ~ x, family = binomial(link = "logit"), data = data)
x.pred <- seq(0, 1, length.out = 300)
y.pred <- predict(glm_model, newdata = data.frame(x = x.pred), type = 'response')
y.pred2 <- ifelse(y.pred > 0.5, 1, 0)

# Summary of the model
summary(glm_model)
```

```
##
## Call:
## glm(formula = y ~ x, family = binomial(link = "logit"), data = data)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -6.3255     0.7577  -8.348   <2e-16 ***
## x            12.6001     1.4553   8.658   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 413.97  on 299  degrees of freedom
## Residual deviance: 153.11  on 298  degrees of freedom
## AIC: 157.11
##
## Number of Fisher Scoring iterations: 6
```

```r
glm_model$coefficients
```

```
## (Intercept)           x
##   -6.325503   12.600096
```

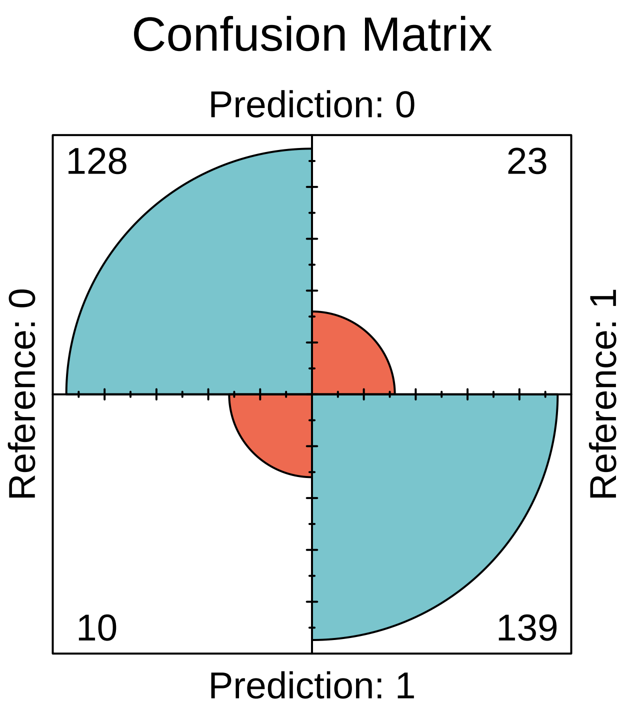
We create a confusion matrix to evaluate the performance of the fitted model.

```r
mat.confusion <- function(predictions, actual_values, main = NULL) {
  cm <- confusionMatrix(predictions, actual_values)

  accuracy <- cm$overall["Accuracy"]
  sensitivity <- cm$byClass["Sensitivity"]
  specificity <- cm$byClass["Specificity"]

  fourfoldplot(cm$table, color = c("coral2", "cadetblue3"), conf.level = 0, main = main)
  return(list(accuracy = accuracy, sensitivity = sensitivity, specificity = specificity))
}

predictions <- factor(y.pred2, levels = c(0, 1))
actual_values <- factor(y, levels = c(0, 1))
mat.confusion(predictions = predictions, actual_values = actual_values, main = 'Confusion Matrix')
```

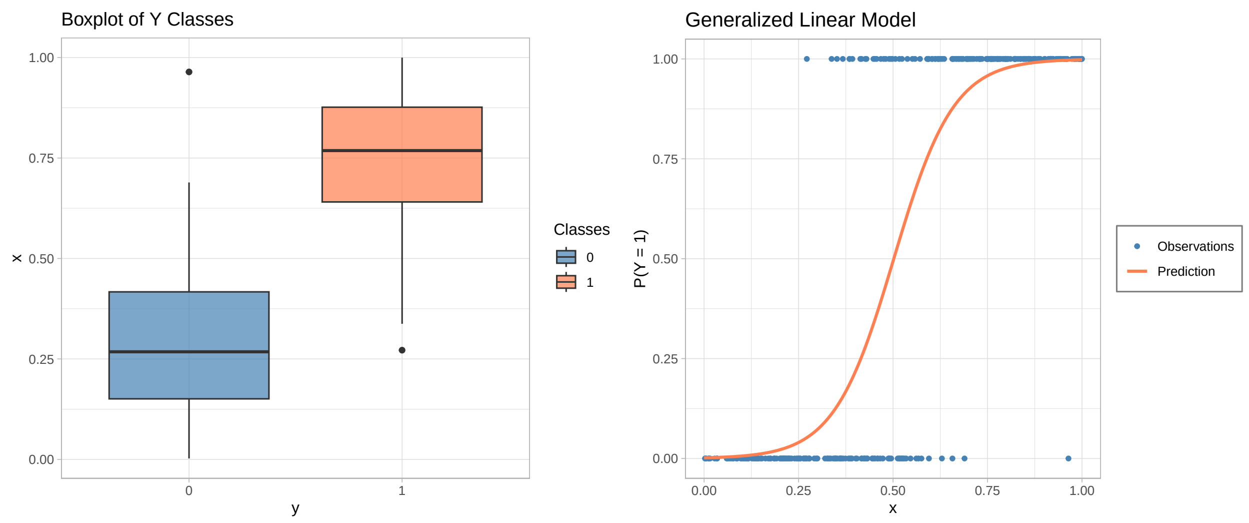


```
## $accuracy
## Accuracy
##     0.89
##
## $sensitivity
## Sensitivity
##   0.9275362
##
## $specificity
## Specificity
##   0.8580247
```

We visualize the fitted GLM along with the observed data and the predicted probabilities.

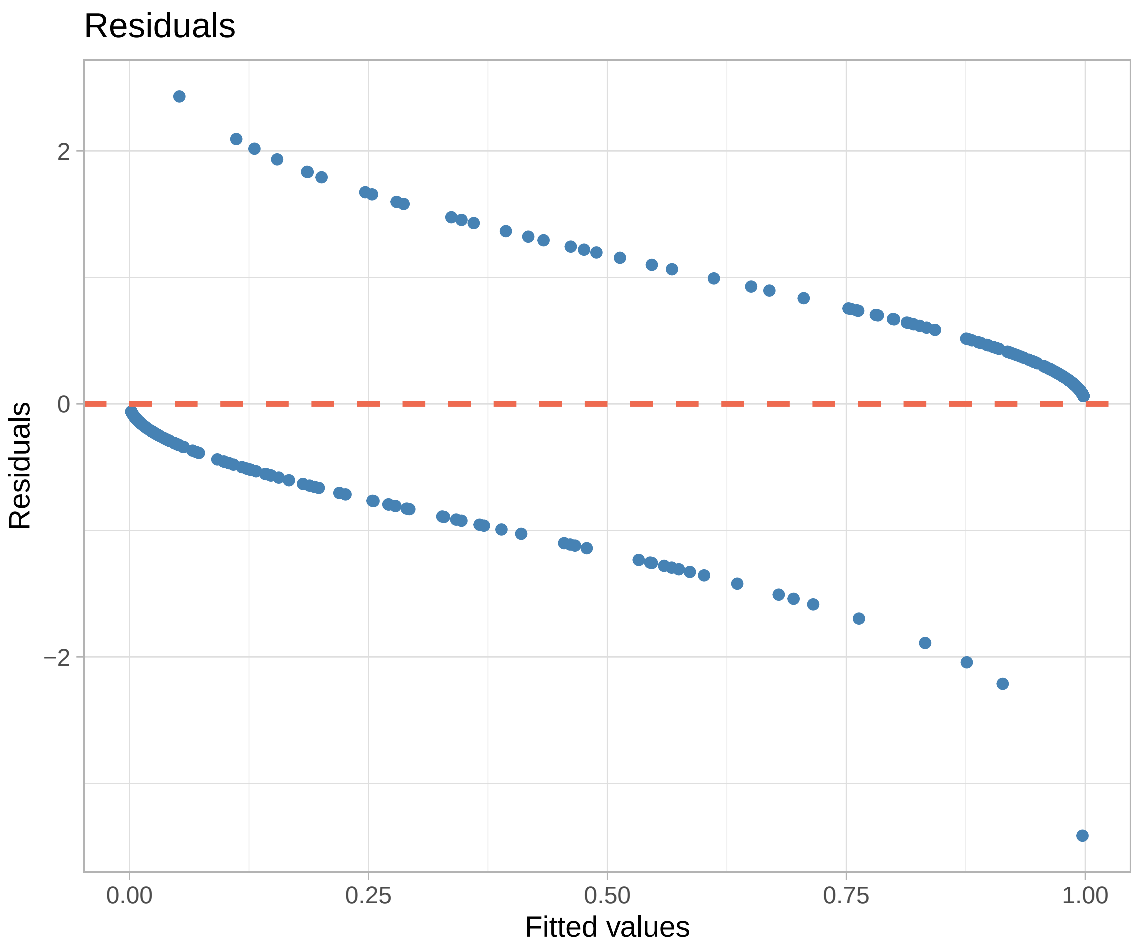
```r
data.model <- data.frame(x, y, prob, x.pred, y.pred)
plot2 <- ggplot(data = data.model) +
  geom_point(aes(x = x, y = y, color = "Observations"), shape = 20, size = 2) +
  geom_line(aes(x = x.pred, y = y.pred, color = "Prediction"), linewidth = 1) +
  scale_color_manual(values = c("Observations" = "steelblue",
                                "Prediction" = "coral")) +
  labs(title = "Generalized Linear Model",
       x = "x",
       y = "P(Y = 1)",
       color = NULL) +
  theme_light() +
  theme(plot.title = element_text(size = 14),
        legend.position.inside = c(0.2, 0.8),
        legend.background = element_rect(fill = "white", colour = "gray50"))

plot <- grid.arrange(plot1, plot2, nrow = 1, ncol = 2)
```



Finally, we perform a residual analysis to assess the model fit.

```r
residuals <- resid(glm_model)
ggplot(data, aes(x = fitted(glm_model), y = residuals)) +
  geom_point(color = "steelblue") +
  geom_hline(yintercept = 0, linetype = "dashed", color = "coral2", linewidth = 1) +
  labs(title = "Residuals",
       x = "Fitted values",
       y = "Residuals",
       color = NULL) +
  theme_light()
```



Manual Parameter Estimation

We manually estimate the parameters of the logistic regression model using the iterative reweighted least squares (IRLS) method.

```r
sigmoid <- function(x) { 1 / (1 + exp(-x)) }

b.i <- c(-5, 10)
X <- cbind(rep(1, n), x)
for (i in 1:100) {
  nu.i <- X %*% b.i
  s <- sigmoid(nu.i)
  W.i <- diag(as.numeric(s * (1 - s)))
  z.i <- nu.i + (y - s) * (1 / (s * (1 - s)))
  b.i <- solve(t(X) %*% W.i %*% X) %*% t(X) %*% W.i %*% z.i
}
print(b.i)
```

```
##        [,1]
##   -6.325503
## x 12.600096
```